```
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
%matplotlib inline
df=pd.read_csv('/content/car_evaluation.csv')
df.head()
```

| | vhigh | vhigh.1 | 2 | 2.1 | small | low | unacc |
|---|---|---|---|---|---|---|---|
| 0 | vhigh | vhigh | 2 | 2 | small | med | unacc |
| 1 | vhigh | vhigh | 2 | 2 | small | high | unacc |
| 2 | vhigh | vhigh | 2 | 2 | med | low | unacc |
| 3 | vhigh | vhigh | 2 | 2 | med | med | unacc |
| 4 | vhigh | vhigh | 2 | 2 | med | high | unacc |

Next steps: Generate code with df View recommended plots New interactive sheet

```
df.shape
```

```
(1727, 7)
```

```
col_names= ['buying','maint', 'doors', 'persons', 'lug_boot', 'safety' ,'class']
df.columns=col_names
col_names
```

```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   buying    1727 non-null   object
 1   maint     1727 non-null   object
 2   doors     1727 non-null   object
 3   persons   1727 non-null   object
 4   lug_boot  1727 non-null   object
 5   safety    1727 non-null   object
 6   class     1727 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

```
df['class'].value_counts()
```

| class | count |
|---|---|
| unacc | 1209 |
| acc | 384 |
| good | 69 |
| vgood | 65 |

```
X = df.drop(['class'] , axis = 1)
Y = df['class']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test,Y_train,Y_test = train_test_split(X , Y ,test_size = 0.33 , random_state = 42)
```

```
X_train.shape,X_test.shape
```

```
((1157, 6), (570, 6))
```

```
!pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl.metadata (8.0 kB)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.26.4)
```

```
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.3.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.13.1)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.14.2)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (2.1.4)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.5.6)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_enco
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (202
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (2
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_encode
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0->category_encode
Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.9/81.9 kB 3.4 MB/s eta 0:00:00
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.3
```

```
import category_encoders as ce
encoder=ce.OrdinalEncoder(cols=['buying','maint', 'doors', 'persons', 'lug_boot', 'safety' ])

X_train=encoder.fit_transform(X_train)
X_test=encoder.transform(X_test)
X_train.head()
```

| | buying | maint | doors | persons | lug_boot | safety |
|---|---|---|---|---|---|---|
| **83** | 1 | 1 | 1 | 1 | 1 | 1 |
| **48** | 1 | 1 | 2 | 2 | 1 | 2 |
| **468** | 2 | 1 | 2 | 3 | 2 | 2 |
| **155** | 1 | 2 | 2 | 2 | 1 | 1 |
| **1043** | 3 | 2 | 3 | 2 | 2 | 1 |

Next steps: Generate code with X_train View recommended plots New interactive sheet

```
from sklearn.tree import DecisionTreeClassifier
clf_gini = DecisionTreeClassifier(criterion='gini',max_depth=3,random_state=0)
clf_gini.fit(X_train,Y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=0)
```

```
Y_pred_gini=clf_gini.predict(X_test)
Y_pred_gini[:5]
```

```
array(['unacc', 'unacc', 'unacc', 'acc', 'unacc'], dtype=object)
```

```
from sklearn.metrics import accuracy_score
print("Model Accuracy score with prediction for test dataset with gini index {0:0.4f}".format(accuracy_score(Y_pred_gini,Y_test)))
```

```
Model Accuracy score with prediction for test dataset with gini index 0.8053
```

```
Y_pred_train_gini=clf_gini.predict(X_train)
Y_pred_train_gini
```

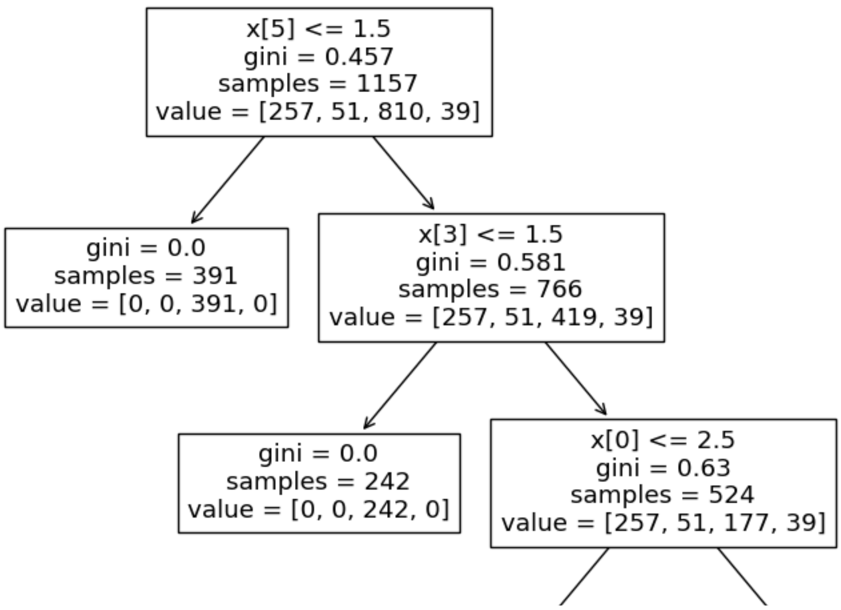
```
array(['unacc', 'unacc', 'unacc', ..., 'unacc', 'unacc', 'acc'],
      dtype=object)
```

```
print("Model Accuracy score with prediction for training dataset with gini index {0:0.4f}".format(accuracy_score(Y_pred_train_gini,Y_tr
```

```
Model Accuracy score with prediction for training dataset with gini index 0.7848
```

```
import matplotlib.pyplot as plt
from sklearn import tree
plt.figure(figsize=(10,8))
tree.plot_tree(clf_gini.fit(X_train,Y_train))
```

```
[Text(0.3333333333333333, 0.875, 'x[5] <= 1.5\ngini = 0.457\nsamples = 1157\nvalue = [257, 51, 810, 39]'),
 Text(0.16666666666666666, 0.625, 'gini = 0.0\nsamples = 391\nvalue = [0, 0, 391, 0]'),
 Text(0.5, 0.625, 'x[3] <= 1.5\ngini = 0.581\nsamples = 766\nvalue = [257, 51, 419, 39]'),
 Text(0.3333333333333333, 0.375, 'gini = 0.0\nsamples = 242\nvalue = [0, 0, 242, 0]'),
 Text(0.6666666666666666, 0.375, 'x[0] <= 2.5\ngini = 0.63\nsamples = 524\nvalue = [257, 51, 177, 39]'),
 Text(0.5, 0.125, 'gini = 0.498\nsamples = 266\nvalue = [124, 0, 142, 0]'),
 Text(0.8333333333333334, 0.125, 'gini = 0.654\nsamples = 258\nvalue = [133, 51, 35, 39]')]
```




```
#Using Gaussian Naive Bias
#The Naive Bayes classifier is a probabilistic model based on Bayes’
#theorem which is used to calculate the probability P(A|B) of an event A occurring, when we are given some prior knowledge B
```

value = [124, 0, 142, 0]] value = [133, 51, 35, 39]]

```
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB(priors=[0.6, 0.3, 0.1, 0.0])

gnb.fit(X_train, Y_train)

print("print Train for accuracy of NBC algo: ", gnb.score(X_train,Y_train))
print("print Test for accuracy of NBC algo: ", gnb.score(X_test,Y_test))
```

```
print Train for accuracy of NBC algo:  0.7519446845289542
print Test for accuracy of NBC algo:  0.7403508771929824
/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py:509: RuntimeWarning: divide by zero encountered in log
  jointi = np.log(self.class_prior_[i])
/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py:509: RuntimeWarning: divide by zero encountered in log
  jointi = np.log(self.class_prior_[i])
```

```
#In summation, Naive Bayes’ independence assumption is a crucial factor for the classifier’s success.
#We have to make sure it applies (to some degree) to our data before we can properly utilize it.
#Likewise, Decision Trees are dependent on proper pruning techniques so that overfitting can be avoided while
#keeping track of the classification objective.
#All in all, they are both very useful methods and a great addition to our toolkit.
```